SYSTEM AND METHOD FOR DETERMINING NETWORK STATUS ALONG SPECIFIC PATHS BETWEEN NODES IN A NETWORK TOPOLOGY

This application is a Continuation-in-Part of U.S. application, Serial Number 09/694,843 filed on October 23, 2000, which is hereby incorporated by reference for all that is disclosed therein.

Field of the Invention

The present invention generally relates to data communication networks and, more particularly, to systems and methods for determining the operational status of devices along a probable network path or paths between two nodes in a network topology.

Background of the Invention

A data communications network generally includes a group of devices, such as computers, repeaters, bridges, switches, routers, *etc.*, situated at network nodes and a collection of communication channels for interconnecting the various nodes. Hardware and software associated with the network and, particularly, the devices permit the devices to exchange data electronically via the communication channels.

The size of networks varies. A local area network (LAN) is a network of devices in close proximity, typically less then one mile, and usually connected by a single cable, for instance, a coaxial cable. A wide area network (WAN) is a network of devices which are separated by longer distances, often connected by, for example, telephone lines or satellite links. In fact, some WANs span the U.S. as well as the world. Furthermore, many of these networks are widely available for use by the public, including universities and commercial industries.

A very popular industry standard protocol for data communication along the networks is the Internet Protocol (IP). In time, the Transmission Control Protocol (TCP) and the Unreliable Datagram Protocol (UDP) were developed for use with the IP. The former protocol (TCP/IP) is a protocol which guarantees transfer of data without errors, as it implements certain check functionality, and the latter protocol (UDP/IP) is a protocol which does not guarantee transfer of data, but requires must less overhead than the TCP/IP platform. Furthermore, in order to keep track of and manage the various devices situated on a network, the simple network management protocol (SNMP) was eventually developed for use with the UDP/IP platform. The use of the foregoing protocols has become extensive in the industry, and numerous venders manufacture many types of networks devices which can employ these protocols.

Many management software packages ("management platforms") are presently available for implementing "management stations" on a network and particularly in connection with the SNMP. Examples of commercially available SNMP management software packages include OpenView from the Hewlett-Packard Company, NetView/6000 from IBM Corp., Spectrum from Cabletron Systems, Inc., Netlabs/Manager from Netlabs, Inc., and SunNet Manager from Sun Connect, Inc. The nodes on a network and their interconnections, oftentimes referred to as a network "topology," are best displayed in a graphical format and most, if not all, of the available management packages provide for this feature. Typically, with these packages, a network can be viewed from different vantage points, depending on the scope of the view desired. For example, one view of the network could be a very wide encompassing view of all nodes on the entire network. A second view could be a view of those portions of a network within a local range, for example, within a particular site or building. A third view of a network, often called a segment, could be a view of the nodes attached to a particular LAN cable.

Hewlett-Packard's very successful OpenView has been the subject of several patents, including, for instance, U.S. Patent No. 5,185,860, issued to

J.C. Wu on February 9, 1993, and U.S. Patent No. 5,276,789, issued to Besaw et al. on January 4, 1994, the disclosures of which are incorporated herein by reference for all that is disclosed therein. U.S. Patent No. 5,185,860 describes an automatic discovery system for a management station for determining the network devices and interconnections of a network, or the topology. U.S. Patent No. 5,276,789 describes a graphic display system for a management station for graphically displaying the topology of a network and provides for various views, including, Internet, segment, and node views, that can be requested by a user.

When utilizing a management station, such as a station implemented by Hewlett-Packard's OpenView, a user may consider it beneficial to be able to predict the probable path or paths between two nodes in a network. Additionally, it may be beneficial to be able to determine the status of data transfer or other network devices situated at the network nodes along the probable path or paths. However, heretofore, the ability to determine the probable path or paths between two nodes has not been addressed by available management platforms. Likewise, the ability to determine the status of network devices along the probable path or paths has not been addressed.

Therefore there is a need for improved systems and methods which address these and other shortcomings of the prior art.

Summary of the Invention

Briefly stated, the present invention relates to determining the status of and continually monitoring network devices along a probable network path or paths between two nodes in a network topology. In this regard, embodiments of the present invention may be construed as providing systems for determining paths between a start node and an end node of a communication network. Typically, such a communication network is formed of sub-networks, with each of the sub-networks including one or more connectors and one or more segments. More specifically, the segments interconnect various ones of the connectors, with the start node, which corresponds to one of the connectors, being designated at one end of the path, and the end node, which corresponds to another of the connectors, being designated at the other end of the path. When the connectors are identified, their statuses are continually monitored.

In a preferred embodiment of the present invention, the system includes a processor, a discovery mechanism, a layout mechanism, and a connector evaluation mechanism. The discovery mechanism is configured to generate and store topology data specifying connectors and segments of a communication network. The layout mechanism, which interfaces with the discovery mechanism, is configured to receive topology data from the discovery mechanism and drive a display based upon the topology data. Additionally, the discovery mechanism is configured to determine a path between a start node and an end node based upon the topology data. For instance, such a path(s) may be based upon information corresponding to a type of path of interest and/or information corresponding to a type of connector of interest. The connector evaluation mechanism is configured to receive parameter values from the connectors along the path representative of their operational status and compare the parameter values to preselected values or specifications. If a connector parameter exceeds a preselected specification, an event is generated signaling an error with the connector.

Some embodiments of the present invention may be construed as providing methods for determining the status of paths between a start node and an end node of a communication network. In this regard, a preferred method includes the steps of: receiving information corresponding to the start node and the end node; receiving information corresponding to a type of path of interest; receiving information corresponding to a type of connector of interest; determining a path between the start node and the end node based upon the type of path of interest and the type of connector of interest; identifying at least one connector in the path; receiving data representative of an operating parameter from the at least one connector; comparing the data to a predetermined value; and providing an indication if the data exceeds the predetermined value.

Other embodiments of the present invention may be construed as providing computer readable media for facilitating a determination of paths between a start node and an end node of a communication network. In this regard, a preferred computer readable medium includes: logic configured to receive information corresponding to the start node and the end node; logic configured to receive information corresponding to a type of path of interest; logic configured to receive information corresponding to a type of connector of interest; logic configured to determine a path between the start node and the end node based upon the type of path of interest and the type of connector of interest; logic configured to receive a parameter value from a connector in the path; logic configured to compare the parameter value to a preselected value; and logic configured to generate an event if the parameter value exceeds the preselected value.

Other features and advantages of the present invention will become apparent to one with skill in the art upon examination of the following drawings and detailed description. It is intended that all such features and advantages be included herein within the scope of the present invention, as defined in the appended claims.

Brief Description of the Several Views of the Drawings

The invention can be better understood with reference to the following drawings. The components in the drawings are not necessarily to scale, emphasis instead being placed upon clearly illustrating the principles of the present invention. Moreover, in the drawings, like reference numerals designate corresponding parts throughout the several views.

FIG. 1 is a block diagram of a management station including discovery/layout software which employs a preferred embodiment of the system and method of the present invention.

FIG. 2 is a schematic diagram illustrating a display map, which comprises a collection of sub-maps, any of which can be displayed on the display of the management station by the discovery/layout software of FIG. 1.

FIG. 3 is a block diagram of the discovery/layout software implementing a preferred embodiment of the present invention.

FIG. 4 is a flowchart depicting functionality of a preferred embodiment of the present invention.

FIG. 5 is a flowchart depicting functionality of a preferred embodiment of the present invention.

FIG. 6 is a flowchart depicting functionality of a preferred embodiment of the present invention.

Fig. 7 is a flowchart depicting functionality of a preferred embodiment of the present invention.

Detailed Description of a Preferred Embodiment

Referring now to the drawings wherein like numerals indicate corresponding parts throughout the several views, FIG. 1 shows a block diagram of an object-oriented management station 100 which is implemented with a general purpose computer system containing novel discovery/layout software 101, which employs a probable path mechanism 103 and associated methodology of the present invention. As shown in FIG. 1, the management system 100 contains a processor 102. The processor 102 communicates with other elements/components within the management station 100 over a system bus 104. An input device 106, for example, a keyboard or mouse, is used to input data from a user of the system 100, and a display 108 is used to output data to the user. A network interface 112 is used to interface the management station 100 to a network 118 in order to allow the management station 100 to act as a node on a network 118. A disk 114, for example, may be used to store the software of the discovery layout software 101 of the present invention, as well as to store the databases (topology and map) generated by the discovery layout software 101. A printer 116 can be used to provide a hard copy output of the nodes of the network 118 discovered by the discovery/layout software 101. A main memory 110 within the management station 100 contains the discovery/layout software 101. The discovery/layout software 101 communicates with a conventional operating system 122 and conventional network software 124 to discover the nodes on the network 118. The network software 124 serves as the intelligence, including validation, for the data communication protocols. As shown in FIG. 1, in the preferred embodiment, the network software implements the IP, the TCP and UDP over the IP, and the SNMP over the UDP. All of the foregoing protocols are well known in the art.

The discovery/layout software 101 implements object-oriented functionality. In the context of SNMP managers, object-oriented means that most of the management system actions and processes that the user can invoke are oriented toward a class of devices rather than individually managed network nodes. Generally, the discovery/layout software 101 of FIG. 1 is configured to discover the network topology, that is, all network node interconnections existing on the network 118, and to construct a map, comprising various sub-maps, any of which can be used for displaying the network topology on the display 108.

Connector evaluation software 120 may also reside within the main memory 110. Two types of connectors are primarily described herein, a layer 3 connector and a layer 2 connector. As utilized herein, a layer 3 (IP layer) connector is defined as a component functioning as a router, *i.e.*, a device that routes data between two or more IP sub-networks (subnets). Additionally, a layer 2 (data link layer) connector is defined as a bridging or switching device which operates within an IP subnet. The connector evaluation software 120 is sometimes referred to as a connector evaluation mechanism. As described in greater detail below, the connector evaluation software 120 causes the management station 100 to communicate with specific connectors within the network 118. The communication causes the connectors to transmit operational parameters back to the management station 100. The connector evaluation software 120 then compares the operational parameters of the connectors to preselected values or specifications. If the operational parameters exceed the preselected specifications, an indication of an error is provided. The indication may include the specific connector that is operating improperly in addition to the parameter and its value that is causing the connector to operate improperly.

FIG. 2 shows a map 200 which is generated by the discovery/layout software 101 from topology data discovered from the network 118. The discovery/layout software 101 can drive any of the various sub-maps to the display 108 (FIG. 1) for viewing by the user. The sub-maps in the map 200 of FIG. 2 are arranged in a hierarchy. A root sub-map 202 is defined at a root level. The root sub-map 202 represents the highest logical level sub-map in the hierarchy and shows objects 203 as anchor points for different sub-map hierarchies. Each hierarchy is a separate management domain. This could be, for instance, a network, logical grouping of nodes, or some other domain. An Internet sub-map 204 is defined at an Internet level and is generated by exploding an object 203 within the root sub-map 202. "Exploding" in the context in this document means that the user prompts the management station 100 with the input device 106 to breakdown and provide more data pertaining to the object 203 at issue. Further, the Internet sub-map 204 illustrates objects 203 in the form of networks and routers. Any one of a number of network sub-maps 206 can be exploded from the Internet sub-map 204. Each network sub-map 206 shows objects 203 in the form of segments and connectors. Any one of a number of segment sub-maps 208 can be exploded from an object 203 within a network sub-map. Each segment sub-map 208 shows objects in the form of network nodes. Finally, any one of a number of node sub-maps 210 can be exploded from an object 203 within a segment sub-map 208. Each node sub-map 210 shows objects 203 in the form of interfaces within that node.

A high level block diagram of the discovery/layout software 101 (FIG. 1) is set forth in FIG. 3 according to a preferred embodiment. With the exception of the probable path mechanism 103, the architecture of the discovery/layout software 101 in FIG. 3 is substantially similar to the architecture of Hewlett-Packard's well known and commercially available management software package called OpenView. As shown in FIG. 3, at a general architecture level, the discovery/layout software 101 comprises a discovery mechanism 302 for discovering nodes and interconnections of the network 118 and a layout mechanism 304 for receiving topology data from the discovery mechanism 302 and for generating the map 200 (FIG. 2) for driving the display 108. Moreover, one or more integrating applications 332 may communicate display and map information with the layout mechanism 304.

The discovery mechanism 302 has a network monitor 306 connected to the network 118 as indicated by connections 308a, 308b, a topology manager 310 connected to the network monitor 306, as indicated by arrows 312a, 312b, and a topology database in communication with the topology manager 310 as indicated by arrow 316. In a preferred embodiment, the probable path mechanism 103 is implemented by or computed within the topology manager 310, as described in detail hereinafter.

The network monitor 306 transmits and receives data packets to and from the network 118. The network monitor 306 discovers and monitors network topology, as indicated by arrows 308a, 308b. When network topology changes on the network, the network monitor 306 generates events, or "traps" (SNMP vernacular), which include an object identifier and object change information. The network monitor 306 can also receive events from other devices, such as a router, in the network 118. The network monitor 306 interacts with the network 118 by way of the network software 124 (FIG. 1), which essentially comprises protocol stacks, corresponding to IP, TCP, UDP, SNMP in the preferred embodiment, and which generally implements these protocols and performs validation functions. Furthermore, the network monitor 306 populates the topology database 314 by way of the topology manager 310 and notifies the topology manager 31 of events (topology changes). Finally, it should be noted that U.S. Patent No.

5,185,860, issued to J.C. Wu, which is incorporated herein by reference, describes a node discovery system which could be employed to implement the network monitor 306 herein.

The topology manger 310, in addition to managing the probable path system and methodology of the present invention, manages the topology database 314, as indicated by bi-directional arrow 316. The topology manager 310 prompts the network monitor 306 to update topology data related to particular events, as indicated by arrow 312a, and receives topology updates, as indicated by 312b.

The topology database 314 stores topology data based upon objects, which are used to partition the network for logical reasons. Objects include, for example, but are not limited to, networks, routers, segments, connectors, nodes, computers, repeaters, bridges, switches, hubs, *etc.* Moreover, the topology data stored with respect to the objects includes, for example, but are not limited to, an interface or device address, an interface or device type, an interface or device manufacturer, and whether an interface or device supports the SNMP.

The layout mechanism 304 has a topology-to-map translator 318 in communication with the topology manger 310 as indicated my arrows 320a, 320b, a graphical user interface (GUI) 322 in communication with the topology-to-map translator 318 as indicated by arrows 324a, 324b, and a map database 326 in communication with the GUI 322 as indicated by bi-directional arrow 328. The integrating application 332 communicates information with the GUI 322, as indicated by arrows 333a, 333b.

It should be noted that the network monitor 306, the topology manager 310, the translator 318, the GUI 322 take turns utilizing the combination of the operating system 122 (FIG. 1) and the processor 102 (FIG. 1) in order to accomplish their respective functions.

The translator 318 converts topology data from the topology database 314 to map data and constructs the various sub maps 202 through 210 in the map 200 of FIG. 2. The translator 318 can forward a request to the topology manager 310, as indicated by arrow 320a, in order to obtain topology data regarding particular objects. Moreover, in addition to forwarding topology data to the translator 318 upon request, the topology manager 310 advises the translator 318, as indicated by the arrow 320b, when topology data has changed based upon an event so that the translator 318 can make any appropriate changes in these submaps.

The GUI 322 manages the map database 326 as indicated by the bi-directional arrow 328, and manages the display 108 and input device 106, as indicated by the arrows 330a, 330b. The GUI 322 receives map updates from the translator 318, as indicated by arrow 324b, and submits user-triggered events to the translator 318, as indicated by arrow 324a. A user-triggered event includes a prompt 330a from a user to explode an object, as described relative to FIG. 2. Finally, it should be noted that U.S. Patent No. 5,276,789, issued to Besaw et al., which is incorporated herein by reference, describes a graphical user interface which could by employed to implement the GUI 322 herein.

Reference will now be made to the flowchart of FIG. 4 which depicts the functionality of a preferred implementation of the probable path mechanism 103 (FIG. 3) and associated methodology of the present invention. In this regard, each block of this flowchart, as well as other flowcharts depicted herein, represents a module segment or portion of code which comprises one or more executable instructions for implementing the specified logical function or functions. It should also be noted that, in some alternative implementations, the functions noted in the various blocks may occur out of the order depicted in the flowchart(s). For example, two blocks shown in succession in FIG. 4 may, in fact, be executed substantially concurrently where the blocks may sometimes be executed in the reverse order depending upon the functionality involved.

As shown in FIG. 4, the preferred functionality depicted may be construed as beginning at block 410 where information corresponding to a start node and an end node of interest may be received. Proceeding then to block 412, information corresponding to a particular path-type of interest may be received. For example, such information may correspond to whether a user is inquiring as to the shortest path and/or all paths between the start node and the end node. Proceeding to block 414, information corresponding to the type(s) of connectors of interest with respect to a given path(s) is received. In particular, such information may correspond to whether layer 2 and layer 3 connectors or, alternatively, only layer 3 connectors are of interest. Thereafter, such as depicted in block 416, the probable path or paths may be determined.

The following definitions are provided for facilitating a more clear understanding of the functionality implemented in the preferred embodiment of the present invention.

"Link" – a connection between an interface on one node and an interface on another node.

"Path" – a conceptual series of links that connect two nodes, *e.g.,* the first interface is on the starting node and the last interface is on the ending node.

"Hop" – a node that serves as the endpoint for a link in the path.

"Hop count" – the number of hops traversed in path, *e.g.*, the first link in a path will have a hop count of zero.

"Shortest path" – a path that traverses the fewest number of hops between the starting node and the ending node.

As mentioned hereinbefore, the present invention may facilitate the determination of probable paths between nodes in a network and may describe the determined path(s) in reference to various types of connectors. More specifically, embodiments of the present invention may define a path as comprising layer 2 and layer 3 connectors or, alternatively, only layer 3 connectors. As utilized herein, a layer 3 (IP layer) connector is defined as a component functioning as a router, *i.e.*, a device that routes data between two or more IP sub-networks (subnets). Additionally, a layer 2 (data link layer) connector is defined as a bridging or switching device which operates within an IP subnet. Thus, when a user requests a probable path determination based upon level 3 connectors only, only connectors operating at level 3 of the network protocol stack will be utilized during path determination. Likewise, when the user requests a probable path determination based upon level 2 connectors and level 3 connectors, connectors operating at both level 2 and level 3 of the network protocol stack will be considered during path determination.

In regard to determining a path or paths that include only level 3 connectors, reference will now be made to the flowchart depicted in FIG. 5 which depicts the functionality of a preferred embodiment of the present invention. As shown in FIG. 5, the functionality represented may be construed as beginning at block 510 where the start node is determined. Determination of the start node may be based upon a designation of a start node by a user. Proceeding to block 512, a determination is made as to what networks communicate with the starting node. Then, as depicted in block 514, each network communicating with the starting node is then evaluated to determine whether the end node, which also may be designated by the user, is associated therewith. In order to promote efficiency during the path determination process, one or more process refinements may be employed. For example, if one of the networks is determined to be an IPX network, that network may be discarded, *e.g.*, appropriately marked so as not to be later utilized during the path determination process. Additionally, if a network has previously been visited, further processing of that network may be avoided, such as by marking that network as "visited."

If it was previously determined, such as in block 514, that the end node is not associated with any of the currently identified networks, the process may proceed to block 516 where identification of a gateway(s) of the identified networks is facilitated. As utilized herein, a gateway is a level 3 connector. For those embodiments incorporating the aforementioned process refinements, it may be assumed that identification of a gateway(s) of a network only is facilitated if the network has not been previously visited and is not an IPX network Thus, when a gateway is identified, a network associated with the gateway also may be identified, such as depicted in block 518. Steps 514 – 518 then may be recursively repeated until one or more paths between the start node and the end node are determined.

When a path between the start node and the end node has been determined during processing, the path may be saved in memory (depicted in block 520). Thus, if a user inquired as to all probable paths between the start node and the end node, each determined path may be saved in memory. Alternatively, if the user inquired as to the shortest path between the start node and the end node, preferably, only the shortest path currently identified by the system during processing is saved in memory. More specifically, if a current shortest path has been previously stored in memory, only information corresponding to a later determined shortest path need be saved in memory, such as by replacing the previous current shortest path. By so doing, an efficient allocation of memory may be achieved.

For ease of description and not for the purpose of limitation, the following segments of pseudo code are provided as representative examples for implementing the above-described functionality.

```
ComputePaths (currentPath, enteringViaNetwork, startingNode, endingNode,
hopCount, currentShortestHopCount, allPaths0rShortestPath,
outputPathList, failedIfsList, loopedOrNot, loopedNodesList,
currentPathNodesList)
{
        If (allPathsOrShortestPath == SHORTEST_PATH &&
currentShortestHopCount <= hopCount)
                return FALSE

        add the startingNode to the currentPathNodesList // Useful during loop
checking
         Create a currentLink with dummy starting and ending interfaces
         Append this link to the outputPath
         Mark startingNode as visited
         Increment hopCount by 1
         Get all the interfaces on this node
         For every interface on this node
         {
```

```
if this interface is marked “not to be searched from”
        continue
get the network to which this interface belongs
if the network is marked visited then
        continue
if transitioning to an IPX network then
        continue
mark this network as visited
Set this interface as the starting interface of the currentLink
Check if any of the endingNode’s interfaces is on this network
If yes then
{
        Set that interface as the ending interface of the current
        link
        If all paths were requested then
                Add this path to the outputPathList
                If total number of paths has reached a maximum
                break
        Else
                Replace the current shortest path with this one on
 the outputPathList
}
else
{
                for every gateway known as currentGateway
 connected to this network
                {
                If gateway is same as the startingNode then
                        Continue
                If gateway is marked visited then
    {
```

```
                Add this gateway to the loopedNodesList if not already
                        present in that list

                        Continue
                    }
                    get the interface of the gateway connected to this
                    network
                    set that as the ending interface of the current link
                    // Make the recursive call to
                    ComputeLevel3OnlyPaths from the next
                    // gateway node
                    call ComputeLevel3OnlyPaths( ...) where ...
                    represents all the required parameters
                    If allPathsOrShortestPath is ALL PATHS then
                        If maximum paths have been obtained
                        then
                            Break
                }
            }
            Clear the visited flag on this network
            If allPathsOrShortestPath is ALL PATHS then
                If maximum paths have been obtained then
                    Break
            Else
                If a shortest path was found
                    Break
            If we did not loop and nor did our descendant from this node
        and if we did not find a path
            then
                Mark this interface of the node as "not to be searched
                from" in the future
```

```
                    Add this interface to the failedIfsList //This list is used to
                    clear this flag in the future on this interface
          }
          Remove the last link from the currentPath
          Clear the visited flag on the startingNode
          Remove this node from the currentPathNodesList
}
```

In contrast to the determination of paths utilizing only level 3 connectors, the determination of paths that include both level 2 and level 3 connectors is facilitated at the segment level (in contrast to the network level determination previously described). In this regard, reference will now be made to the flowchart of FIG. 6 which depicts the functionality of a preferred embodiment of the present invention. As shown in FIG. 6, the functionality represented may be construed as beginning at block 610 where the start node is determined. Proceeding to block 612, a determination is made as to what segment(s) communicate with the start node. Then, as depicted in block 614, each segment communicating with the start node is then evaluated to determine whether the end node is associated therewith. As described hereinbefore (in regard to the flowchart of FIG. 5), efficiency of the path determination process may be enhanced by one or more process refinements, *e.g.*, disregarding nodes associated with an IPX network, among others.

If it was previously determined, such as in block 614, that the end node is not associated with any of the currently identified segments, the process may proceed to block 616 where identification of connector(s) and/or gateway(s) associated with the identified segments is facilitated - a gateway is a level 3 connector. Then, as depicted in block 618, segments associated with the newly identified connector(s) and/or gateway(s) may be evaluated. Steps 614 – 618 may be recursively repeated until one or more paths between the start node and the end node are determined. It should be noted that, when transitioning from one segment to another segment during processing, a segment which belongs to another network is not utilized unless the connector associated with that segment is a gateway. This is because level 2 connectors connect segments within a particular subnet and only gateways connect different subnets. Conversely, transitioning between a segment of one network or subnet and a segment of another network or subnet only occurs at a gateway.

When a path between the start node and the end node has been determined during processing, the path may be saved in memory (depicted in block 620). Thus, if a user inquired as to all probable paths between the start node and the end node, each determined path may be saved in memory. Alternatively, if the user inquired as to the shortest path between the start node and the end node, preferably, only the shortest path currently identified by the system during processing is saved in memory, such as described hereinbefore in relation to the flowchart of FIG. 5.

In regard to the computation of paths (paths designated by both layer 2 and layer 3 connectors) the following representative pseudo code is provided for ease of description and not for the purpose of limitation.

```
ComputePaths (currentPath, enteringViaNetwork, startingNode, endingNode,
hopCount, currentShortestHopCount, allPaths0rShortestPath,
outputPathList, failedIfsList, loopedOrNot, loopedNodesList,
currentPathNodesList)
{
        If (allPathsOrShortestPath == SHORTEST_PATH &&
                currentShortestHopCount <= hopCount)
                return FALSE

        add the startingNode to the currentPathNodesList // Useful during loop
checking
        Create a currentLink with dummy starting and ending interfaces
        Append this link to the outputPath
        Mark startingNode as visited
```

```
Increment hopCount by 1
Get all the interfaces on this node
For every interface on this node
{
        if this interface is marked "not to be searched from"
                continue
        get the segment to which this interface belongs
        if the segment is marked visited then
                continue
if transitioning to an IPX network then
continue
        if startingNode is not a gateway and if the network to which this
interface belongs is
        different from the enteringViaNet then
                continue // Can not transition to another network unless
                device is a gateway
        mark the current network as visited
        mark the current segment as visited
        Set this interface as the starting interface of the currentLink
        Check if any of the endingNode's interfaces is on this segment
        If yes then
 {
        Set that interface as the ending interface of the current link
                If all paths were requested then
                        Add this path to the outputPathList
                        If total number of paths has reached a maximum
                                break
                Else
                        Replace the current shortest path with this one on
                the outputPathList
        }
        else
        {
        for every connector known as connector connected to this
    network
    {
                    If connector is same as the startingNode then
                    Continue
                    If connector is marked visited
                    then
        }
                        Mark that we looped here
                        Add connector to the loopedNodesList if
                    not already present
                        in that list
                        Continue
        }
                    get the interface of the connector connected to
                this network
                    set that as the ending interface of the current link
                    // Make the recursive call to ComputePaths from
                the next
                    // gateway node
                    call ComputePaths(...) where ... represents all the
                required
                    parameters
                    If allPathsOrShortestPath is ALL_PATHS then
                        If maximum paths have been obtained
                    then
                            Break
    }
    }
      Clear the visited flag on this segment
      If startingNode is a gateway then,
                Clear the visited flag on this network
            If allPathsOrShortestPath is ALL_PATHS then
                If maximum paths have been obtained then
                Break
            Else
            If a shortest path was found
                Break
            If we did not loop and nor did our descendant from this node
        and if we did not find a path
            then
            Mark this interface of the node as "not to be searched from" in
        the future
                Add this interface to the failedIfsList //This list is used to
                clear this flag in the future on this interface
        }
        Remove the last link from the currentPath
        Clear the visited flag on the startingNode
            Remove this node from the currentPathNodesList
    }
```

When the paths and connectors have been identified as described above, the connectors within the paths may be monitored to assure that they are functioning properly. By first determining a probable path or paths between nodes of interest, only the connectors along the probable paths need to be monitored. This reduces the time and resources required for monitoring a network because only specific connectors need to be monitored.

This process of monitoring the connectors is illustrated by the flowchart of FIG. 7, which begins in a manner similar to the functionality depicted in the flowchart of FIG. 4. As shown in FIG. 7, the preferred functionality depicted may be construed as beginning at block 710 where information corresponding to a start node and an end node of interest may be received as was described above. Proceeding then to block 712, information corresponding to a particular path-type of interest may be received. For example, such information may correspond to whether a user is inquiring as to the shortest path and/or all paths between the start node and the end node. Proceeding to block 714, information corresponding to the type(s) of connectors of interest with respect to a given path(s) is received. In particular, such information may correspond to whether layer 2 and layer 3 connectors or, alternatively, only layer 3 connectors are of interest. Thereafter, such as depicted in block 716, the probable path or paths may be determined.

When the probable path or paths have been determined, a connector in the path is identified as depicted in block 718. Proceeding to block 720, information relating to an operating parameter is received from the connector. With additional reference to Fig. 1, the level two and level three connectors in the network 118 may have the ability to communicate with the management station 100. For example, the network 118 may use the SNMP, which provides for communications between the management station 100 and the connectors. In addition, the level two and level three connectors may store operational parameters locally. The operational parameters may, as a non-limiting example, be stored within the connectors in the form of conventional management information bases (MIBs). The MIBs may include such parameters as the traffic passing through the connector, the disc space available on the connector, and any errors in the operation of the connector that have been detected by self diagnostics.

The values of the above-described parameters are then transmitted or otherwise sent to the management station 100. The connector evaluation software 120 receives the values of the connector parameters and compares them to preselected values or specifications as is depicted in block 722. Should a value of a connector parameter exceed a preselected value or specification, as depicted in block 724, an alarm or other indication may be provided to notify a network administrator or the like of the problem as is depicted in block 726. The connector evaluation software 120 may continually operate in order to find errors before they become significant. For example, the connector evaluation software 120 may periodically retrieve the above-described connector parameters from the connectors for evaluation. Accordingly, the connector evaluation software 120 may continually evaluate the status of the connectors by continually comparing the connector parameters to the above-described connector values.

The above-described specifications within the connector evaluation software 120 may be established by various methods. In one embodiment, a probable path is found as described above. A user may then view the connectors located within the path and set the parameter specifications for each of the connectors. In another embodiment, the parameter specifications may be set for each type of device in the path.

In another embodiment, the parameter specifications are preestablished for each of the different connectors that may be located in the path. For example, a specific type or model of connector may have specific parameter specifications associated therewith. This allows for ease in establishing the parameters specifications by simply associating the specifications with the type of connector that is found to be in the path. Another method for establishing the parameter specifications is to select them based on the operational history of the network as is described in the United States patent application serial number 09/915,070 of Loyd filed on July 25, 2001 for METHOD AND DEVICE FOR MONITORING THE PERFORMANCE OF A NETWORK (attorney docket number 10006615-1), which is hereby incorporated by reference for all that is disclosed therein.

The foregoing description has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Modifications or variations are possible in light of the above teachings. The embodiment or embodiments discussed, however, were chosen and described to provide the best illustration of the principles of the invention and its practical application to thereby enable one of ordinary skill in the art to utilize the invention in various embodiments and with various modifications as are suited to the particular use contemplated. All such modifications and variations are within the scope of the invention as determined by the appended claims when interpreted in accordance with the breadth to which they are fairly and legally entitled.